

Counting Sort.

Note Title

11/8/2007

// input must be nonnegative ints
// A is input, B is output

// for each input $A[i]$, find
// all values less than it.

⑤ Let $MaxVal = \max(A)$;

① Initialize all $C[0 \dots MaxVal]$ to zero

② For $j = 1$ to n

$C[A[j]]++$;

// $C[i]$ is # elements in A equal to i

④ For $i = 1$ to $MaxVal$

$C[i] = C[i] + C[i-1]$.

// Now $C[i]$ is # elements in $A \leq i$

⑥ For $k = n$ to 1

Let $key = A[k]$. // value in A

Let $count = C[key]$. // # elements $\leq val$

$B[count] = key$;

$C[key] = count - 1$;

Counting sort example

Input A:

2	1	6	5	2	5	3	1
---	---	---	---	---	---	---	---

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

⑤ $\max \text{Val} = \max(A) = 6$

① Initialize all $C[0 \dots b]$ to \emptyset

C

\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
-------------	-------------	-------------	-------------	-------------	-------------	-------------	-------------

\emptyset	1	2	3	4	5	6
-------------	---	---	---	---	---	---

② Let $C[i] = \# \text{elements in } A$
equal to i

C

\emptyset	2	2	1	\emptyset	2	1
-------------	---	---	---	-------------	---	---

\emptyset	1	2	3	4	5	6
-------------	---	---	---	---	---	---

④ Let $C[i] = \# \text{elements in } A \leq i$

C

\emptyset	2	4	5	5	7	8
-------------	---	---	---	---	---	---

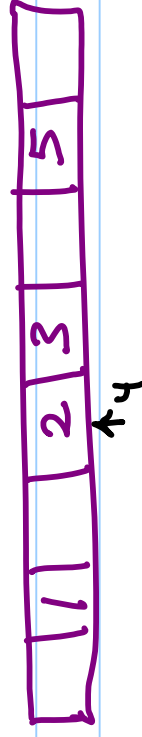
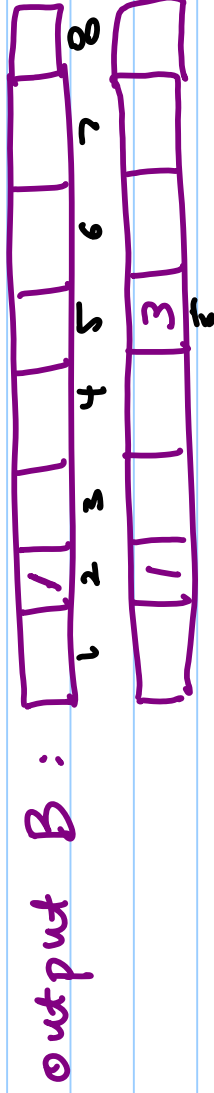
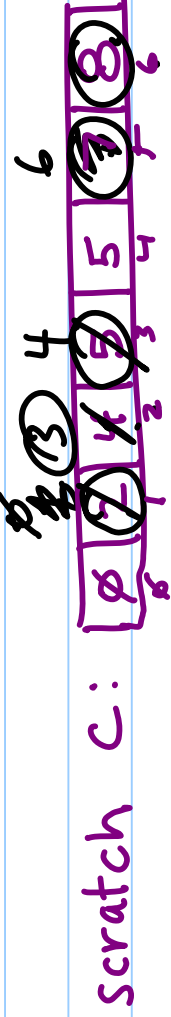
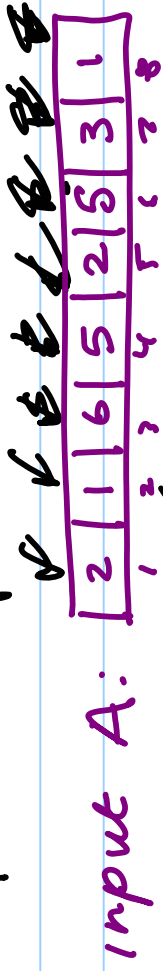
⑥ Let $B[1 \dots n]$ be the OUTPUT array.
Note: if $C[i]$ has the # elements $\leq i$,

then $B[C[i]] = i$

count of all values $\leq i$

example, if $C[3] = 4$,
then there are 4 values ≤ 3 ,
so put a 3 in the 4th slot in B.

Work backwards from rightmost input element to leftmost input element.



Counting Sort

What if we also have negative integers?
or $\min > \phi$?

- compute Min
- compute Max.

- give C size $(\text{Max} - \text{Min})$.
 $C[\phi \dots (\text{Max} - \text{Min})]$.

- subtract min from all values
- do Counting Sort as before.

- add min to all elements in B.

alternatively, subtract min from values used as indices to C.

Runtime of Counting Sort (array reads,
array writes)

$$\Theta(\max(n, \text{MaxVal}))$$

or

$$\Theta(\max(n, \underbrace{\text{range}}_{\text{max} - \text{min} + 1}))$$

equivalent

$$= \Theta(n + \text{range})$$

if range is $O(n)$,

then algorithm is $\Theta(n)$.

Radix Sort for nonneg. integers

Sort one digit at a time

329	457	329	329
457	657	839	457
657	329	457	657
839	839	657	839

sort
rightmost
digit first

sort
leftmost
digit last

Digit sorts must be STABLE. counting sort is a good choice.

— MaxVal on digit sort is 9.

so digit sort is $O(\max(n, 9)) \rightarrow O(n)$.

— so runtime of radix sort is $O(dn)$
 $d = \# \text{ digits}$

Σ

General Radix Sort Runtime

$d = \#$ "digits"
(could be other data)

$r =$ range of each digit

$n = \#$ values.

Radix sort runtime is $O(d(n+r))$

(equivalent to $O(d \cdot \max(n, r))$)

if d is fixed and $r \in O(n)$,
then radix sort is linear.

Question 1: If we have n b -bit integers, can we sort them in $\Theta(b \cdot n)$ time?

Use radix sort.

digits = b

range = 2 (\emptyset or 1)

$n = n$

so runtime is $\Theta(\overset{\#}{\text{digits}}(n + \text{range}))$

$$= \Theta(b(n+2))$$

$$= \Theta(bn)$$

Question 2: How many bits are used to represent the numbers in the range $0 \dots n-1$?

<u>n</u>	<u>n-1</u>	<u># bits needed</u>	
4	3	2	01 10 11
8	7	3	001 010 011 100 101 110 111
16	15	4	0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

$\log_2 n$ bits!

So if sorting n b -bit integers takes

$\Theta(bn)$ time, then sorting

$n (\log_2 n)$ -bit integers takes $\Theta(n \log n)$ time!

Question 3: What if we group the digits into clusters of size r ?

2317	→	2317	0896	
6542		4321	1239	
4321		1239	→	2317
0896		6542		4321
1239		0896		6542

$r = 2$ ↙ # of decimal digits

"digits" = $2 = 4/r$

$$\text{range} = 100 = 10^2 = 10^r$$

$$\text{so runtime} = \Theta(2(n + 100))$$

$$= \Theta\left(\frac{4}{r}(n + 10^r)\right)$$

If n b -bit digits clustered into clusters of size r ,

$$\text{runtime is } \Theta\left(\frac{b}{r}(n + 2^r)\right)$$

Radix sort one bit at a time
is

$$\Theta(b(n+2)) = \Theta(bn)$$

Radix sort r bits at a time is

$$\Theta\left(\frac{b}{r}(n+2^r)\right) = \Theta\left(\frac{b}{r}n\right)$$

Save time, but counting sort now
needs more space...
(MaxVal is bigger when $r > 1$)

In decimal example, MaxVal
grew from 9 to 99

Claim: Given the number of bits to represent n numbers is $O(\log n)$, then if we do all of the bits in a single grouping, Radix Sort runs in $\Theta(n)$ time.

$$b \in O(\log n)$$

let $r = \log_2 n$ (ie, # of bits)

$$\text{Radix Sort is } \Theta\left(\frac{b}{r} (n + 2^r)\right) \quad \frac{b}{r} = 1$$
$$\Theta(n + 2^r)$$
$$\Theta(n + 2^{\log_2 n})$$
$$\Theta(n + n)$$
$$\Theta(n)$$

Lots of hidden constants + memory.

Can we sort n values that are in the range $\phi \dots n^2$ in $O(n)$ time?

n values in the range $0 \dots n^2$

ok, let's try regular radix sort

#digits = d , $d \leq n^2$

runtime = $\Theta(d(n+r)) = \Theta(d(n+n^2))$
eek!

Another way:

- represent all values in binary
- split binary number in two

example: 2 8 6 18 10 36

decimal

binary

$n=6$
 $n^2=36$

2 000010
8 001000
6 000110
18 010010
10 001010
36 100100

SORT
 000010
 001000
 000110
 010010
 001010
 001010
 100100
 100100
 SORT
 001000 2
 000110 6
 001000 8
 001010 10
 010010 18
 100100 36

"digits" = 2 (# clusters)
max Val = 111 (binary) or $7 \leq 2n$

So runtime

$$\text{is } \Theta(n+r) = \Theta(2(n+2n))$$

$$= \Theta(n)$$

ch. 8

Bucket Sort

Linear expected time.

Assume elements equally distributed across the range of values.

Create n buckets.

example: numbers between ϕ (inclusive) and 1 (exclusive) $[\phi, 1)$

bucket 1: $[\phi, \dots, \frac{1}{n})$

bucket 2: $[\frac{1}{n}, \dots, \frac{2}{n})$

bucket n : $[\frac{n-1}{n}, \dots, \frac{n}{n})$

if bucket 1 is empty, then bucket 2 is empty, etc.

example: numbers between ϕ and 1000

bucket 1: $[\phi, \dots, \frac{1000}{n})$

bucket n : $[\frac{(n-1)(1000)}{n}, \dots, 1000)$

Bucket Sort ($A[1 \dots n]$) {

④ Determine RANGE of values
using MAX and MIN

① Create n equal-sized
buckets based on RANGE

② for $i = 1$ to n
insert $A[i]$ into
corresponding bucket

⑤ Sort each bucket

⑥ Concatenate buckets

Runtime: MAX, MIN: $\Theta(n)$

put in buckets: $O(n)$

concatenate buckets: $O(n)$

BUT SORT EACH BUCKET?

How long to sort each bucket?

example: 8 3 2 \emptyset

range: $\emptyset \rightarrow 8$

$n = 4$

	Bucket 1	Bucket 2	Bucket 3	Bucket 4
	$[\emptyset - 2)$	$[2 - 4)$	$[4 - 6)$	$[6 - 8]$

\emptyset	3, 2	—	8
-------------	------	---	---

sort	\emptyset	2, 3	8
------	-------------	------	---

answer	\emptyset	2	3	8
--------	-------------	---	---	---

note: # elements in each bucket
very low!!

b/c of our ASSUMPTION of uniform
data values!

b/c of our ASSUMPTION of uniform
data values:

expected # of
values per bucket is very small.

example: if max bucket size is 8,
then # comparisons to sort = $\frac{24}{8}$.
sort time is $\frac{24}{8} \times n \text{ lists} = 24n$.
Total runtime is $\Theta(n)$.

